Learning and Adaptation in an Airborne Laser Fire Controller

Phillip D. Stroud (stroud@lanl.gov)
Los Alamos National Laboratory
Mail Stop F607, Los Alamos, NM 87545

LAUR 95-1442
IEEE Transactions on Neural Networks, Vol. 8, No. 5, 1997

Abstract-- A simulated battlefield, containing airborne lasers that shoot ballistic missiles down, provides an excellent test-bed for developing adaptive controllers. An airborne laser fire controller, which can adapt the strategy it uses for target selection, is developed. The approach is to transform a knowledge-based controller into an adaptable connectionist representation, use supervised training to initialize the weights so that the adaptable controller mimics the knowledge-based controller, and then use directed search with simulation-based performance evaluation to continuously adapt the controller behavior to the dynamic environmental conditions. New knowledge can be directly extracted from the automatically discovered controllers. Three directed search methods are characterized for production training, and compared with the better characterized gradient descent methods commonly used for supervised training. Automated discovery of improved controllers is demonstrated, as is automated adaptation of controller behavior to changes in environmental conditions.

Keywords: Adaptive control, simulation, complex system, evolutionary programming, airborne laser, neural network.

1. Introduction

Simulation is playing an increasingly important role in training, analysis, and operational planning for many complex systems. In the powerful distributed object-oriented simulation approach, software actors emulate the various system elements, and interact through asynchronous message passing [1]. An important class of complex system contains entities which can adapt their behavior to make themselves more effective within their complex, dynamic environment [2-4]. These entities have a component which perceives the state of the entity and its environment, and selects an action from a set of possible actions. The behavior of an entity is its mapping from the space of all possible perceived states onto the set of all actions the entity is capable of performing [5]. When an entity continuously selects its own actions, the behavior forms the basis of a controller for the entity.

This paper begins by developing a traditional knowledge-based controller to select targets for an Airborne Laser (ABL) theater missile defense system. The fire controller inputs are the parameters associated with possibly dozens of engageable boosting missiles. The fire controller output specifies which of the missiles to intercept next. This knowledge-based controller incorporates the expertise of mission analysts and design engineers, and can be encapsulated in the rule: *select that target which can be destroyed in the shortest time*. Section 2 describes the ABL, the environment in which it operates, the role of the fire controller, and the knowledge-based expert controller.

This knowledge-based fire controller, though it performs much like a human operator, is static. Furthermore, it arose from experience with a limited set of scripted scenarios. Theater missile defense occurs in a complex system that contains many interacting elements (transporter/launchers, missiles, radars, surveillance and combat aircraft, air and ground based interceptors, troop units, command centers, weather, etc.). Tactics can evolve on both sides. For example, new spatial or temporal missile launch patterns may be found to be more effective against a baseline missile interceptor doctrine. In this complex environment, a controller must be able to adapt its behavior. The software controller of a simulated entity in a simulation must likewise have the ability to adapt its behavior.

The ability to adapt to a dynamic environment is achieved by transforming the knowledge-based controller into an adaptable connectionist representation, and then using directed search methods to find new behaviors that are more effective [6]. The directed search Evaluation of the performance of trial controllers, which is fundamental to any search, is accomplished by running simulations of the complex battlefield. A connectionist configuration, the multi-linear network [7], provides an adaptable representation of the fire controller. The multi-linear network representation of the fire controller is described in Section 3.

There are two types of training that apply to adaptable controllers. The first type, supervised training, is a search for the connectionist controller that best mimics a given state-action mapping (in this case, that of the knowledge-based controller). The supervised training process is described in Section 4. The second type, production training, is a directed search for improved state-action mappings, using simulation-based evaluation of how well an actor achieves goals in its interactions with the rest of the system. Real-time or on-line adaptation can be accomplished if the directed search process is carried out faster than the time scale of the changes in the system. The production training process is described in Section 5.

Three classes of training methods, based on regression, gradient descent, and directed search, are examined in this paper. Regressive methods provide extremely efficient supervised training of the multi-linear network. They are used to find the optimal solution, which is useful in evaluating the gradient descent and directed search methods. The regressive training methods are examined in Section 6.

Gradient descent methods are used in most applications of supervised neural network training, and as such are well characterized. They are not generally effective in production training, since there will be many local optima, and since the gradient can not be implicitly obtained. Gradient descent methods can provide a standard of comparison for the directed search methods. The use of gradient descent methods for supervised training of a multi-linear network ABL fire controller is developed in Section 7.

Directed search methods can be applied to both supervised and production training. Three directed search methods are examined in Section 8: forward-biased pivot offset with simulated annealing, downhill simplex, and genetic algorithms. Each of these methods is compared with the gradient descent methods for supervised training, based on the required number of performance evaluations. These directed search methods are also evaluated for production training. Automatic discovery of improved controllers is demonstrated. Several examples are provided in which the fire controller adapts its behavior to modifications in environmental conditions.

2. The Airborne Laser Fire Controller

2.1 The Airborne Laser theater ballistic missile interceptor.

The ABL is a system that intercepts theater ballistic missiles during their boost phase [8,9]. It consists of a dedicated wide-body aircraft, a high power laser, an optical train culminating with a turret mounted transmitting mirror, and a suite of sensors and missile trackers. Since the reach of the system is several hundred kilometers, these intercepts can take place over enemy territory while the ABL itself remains in friendly airspace. The system thus can protect large areas and troop formations from missiles capable of carrying any of a variety of munitions. In operation, an ABL flies out to a designated region of airspace and begins a surveillance pattern. As missiles are detected, they are entered into a track file maintained on the ABL. The laser can fire at one missile at a time.

When there are multiple missiles in the track file, the overall performance of the ABL depends on how intelligently it chooses the target engagement order. Because of uncertainty in the burn-out time of the missiles, the merit of a particular target selection order contains a stochastic component and can not be definitely evaluated. There is some likelihood that a missile will burn out during ABL engagement, thus surviving to deliver its payload. As the number of missiles in the track file gets large, the number of permutations in the firing order quickly becomes too large to evaluate all of the possibilities. The ABL fire controller has the task of selecting which missile (out of possibly dozens in the track file) to engage next, in the presence of uncertainty, and without the possibility of evaluating the exact merits of the choices.

2.2 The knowledge-based ABL fire controller.

A knowledge-based fire control strategy emerged during the conceptual development of the ABL. This baseline expert fire controller selects that target in the track file which is evaluated to require the shortest amount of time to destroy. The destruction time estimate is the sum of the slew time (to rotate the turret so it points toward the target) and the dwell time (to accumulate a lethal fluence with the deliverable intensity). The deliverable intensity is a complicated function of a number of parameters: the range to target, the target altitude, the azimuth angle from the nose of the plane to the target, the target aspect angle, target and platform velocities, atmospheric conditions, etc. This expert fire controller requires high fidelity atmospheric propagation and missile engagement models, so that it can accurately evaluate the deliverable intensity [10-12]. The priority of a target is taken as the inverse of its estimated time-to-kill. The controller evaluates the target priority value for each missile in the track file, and selects that missile with the highest value. The target priority as a function of range to target and target altitude is shown in Fig. 1, for targets on bore-sight, with representative values taken for the remaining parameters. The ridge structure that appears between 10 and 15 km target altitude is due to a layer of strong atmospheric turbulence located at the tropopause. This knowledge-based expert controller has been used in several major ABL simulations

[13-15], and was able to perform at about the same level as a human operator manually selecting targets on a simulated ABL console [16].

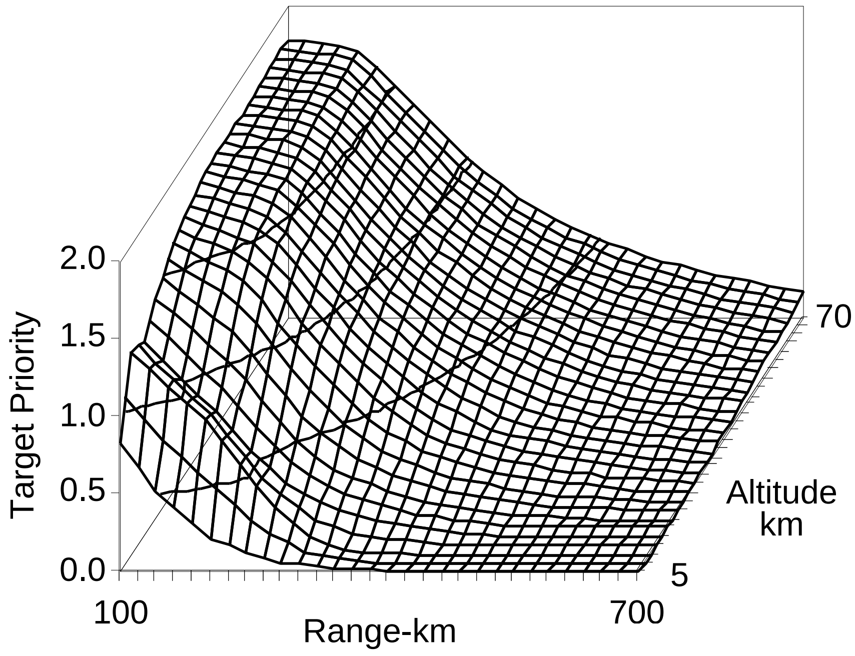

Figure 1. The target priority assigned by the shortest time-to-kill knowledge-based fire controller, as a function of the range to target, and the target altitude, for targets on bore-sight (no slew), and other target parameters fixed at representative values.

2.3 Simulation-based evaluation of the fire controller performance.

A simulation package ELASTIC (Evolutionary Los Alamos Simulation-based Training for Intelligent Controllers) was developed to evaluate the performance of various controllers. ELASTIC provides a collection of C++ classes that model the various entities in missile defense. The kernel of the simulation evaluates the number of missiles killed in a sortie. In a sortie, an airborne laser flies to its designated loiter area and begins surveillance. Theater missiles are launched at unknown times, from unknown launch locations, and to unknown targets. The launch zone, target zone, number and type of missiles are selected for consistency with a scenario of interest. The scenario used here has a rectangular missile launch zone, 400 km wide and 200 km deep. The ABL is restricted to a rectangular area 240 km wide and 50 km deep. The launch zone and loiter box are 200 km apart. The missiles are launched in the general direction of the loiter box, with an angular spread of ±25°. ELASTIC generates random missile launch scripts consistent with the scenario. The missiles are flown with a 4 degree-of-freedom trajectory model, in order to obtain sufficient fidelity. The ABL system parameters (laser power, beam transmitter aperture, etc.) were selected to provide roughly a one-half or two-thirds intercept probability.

A nominal salvo was arbitrarily fixed at 100 missiles launched within a three minute period. The missiles have a fly-out range of between 400 and 600 km. In order to obtain statistical significance, ten independent salvos are simulated for the evaluation of each trial controller, for a total of 1000 missiles. The simulation has been implemented on several computers. On a Macintosh Quadra, 90.0 sec of computing time are required to simulate these 10 sorties and get a performance evaluation of a trial controller. On a DEC alphaStation, an evaluation requires 1.71 sec, while 1.95 sec are required on an HP Apollo series 735 work station.

The same 10 sorties are used to evaluate the various controllers. The shortest time-to-kill fire controller intercepts 625 out of these 1000 missiles. For comparison, a fire controller that selects the next target at random only intercepts 512 of these 1000 missiles. A case was simulated in which the missiles were launched with long time intervals, so that there was never more than one target for the fire controller to choose from. In this case, the ABL intercepts 750 out of 1000 missiles. For this particular geometry, 250 of the 1000 missiles are not engageable regardless of the fire controller, because they are too far away, or because the plane is heading the wrong way when they are launched.

The accuracy of a performance evaluation based on simulation of 1000 missile engagements is characterized as follows. Assume that 250 of the 1000 missiles are unengageable. Of the remaining

750 missiles, 625 are intercepted in a particular evaluation. The intercept probability for engageable missiles is then estimated at p=0.833. The unbiased estimator of the standard deviation [17] of the number of missiles intercepted out of the 750 engagable would be given by $750[p(1-p)/749]^{1/2}$ = 10.2. The estimation of the performance of the controller after a simulation of 1000 engagements is thus 625 ± 10.2. A series of 20 independent simulations of 1000 engagements each, using the shortest time-to-kill controller, had an actual standard deviation of 9.3 missile intercepts.

There are several obvious approaches for improvement of this baseline fire control algorithm. One such approach would employ an estimate of the probability that a missile receives a lethal fluence prior to burning out. The target selection would then be based on maximizing the kill probability per unit time. In a chess-like approach, all targets in the track file would be evaluated for estimated time-to-kill or kill probability per unit time. The best three or four might then be evaluated at higher fidelity, looking at all firing order permutations. There is also information about missile clustering that might be used. These manual knowledge engineering approaches to improving the fire controller are not pursued in this paper. Instead, the baseline fire controller is transformed into an adaptable structure, which then learns better behavior automatically.

3. The Multi-linear Net

The knowledge-based fire controller described above provides a mapping from several target parameters into a target priority value, for each missile in the track file. The target priority was found to depend strongly on just three parameters: the range to target, the target altitude, and the required turret slew angle. A starting guideline for designing the structure of an adaptable representation of the fire controller is that it takes these three parameters as inputs, and generates an output that characterizes the priority of the target. The controller input $\{x_k\}$ is a vector of these three target parameters, roughly normalized onto the interval (0,1). x_1 represents the ground range from the ABL to the target. x_2 represents the target altitude above the ground. x_3 is the normalized angle from the current beam turret bearing to the target bearing. The input parameters are summarized in Table 1. The actual parameters (range, altitude, slew angle) map linearly onto the controller input parameters, $\{x_k\}$. A missile altitude of 25km, for example, corresponds to x_2= 0.308.

parameter		interpretation of small value (0)	interpretation of large value (1)
x_1	range to target (km)	near 100 km	far 700 km
x_2	target altitude (km)	low 5 km	high 70 km
x_3	slew angle (deg)	on bore-sight 0 deg slew	off bore-sight 120 deg slew

Table 1. The ABL fire controller input parameters

An adaptable structure that assigns an adjustable priority value to various regions of the input space is capable of representing the knowledge contained in the original controller. A multi-linear expansion, implemented as a connectionist network, is ideally suited for representing knowledge with this structure [8]. The multi-linear net is a special case of a functional link net [18], in which the expansion functions are localized in the input space, rather than formed by various polynomials. It can also be seen as a non-radial variant of a radial basis function expansion network. It can be considered to be a three layer network. The first layer has one input neuron for each of the three target parameters. The input neurons can perform the normalization of the input parameters, or simply receive pre-normalized parameters. Each neuron in the second layer corresponds to a particular region of input space. The second layer has one neuron per expansion function, the output of which indicates to how close the input state is to the corresponding region of input space. If the input vector matches one of the expansion function centers exactly, the corresponding node will have a value of 1, while all other second layer nodes will have a value of 0. The third layer is an output node which produces a linear combination of the second layer outputs. The weights of the output layer are adjustable, and have a semantic interpretation as the target value of the corresponding regions of input space.

Any linear transformation of one input x to output y can be written $y = w_1(1-x) + w_2 x$. This can be seen as a linear combination of two functions of x, namely h_1=1-x and h_2=x. The expansion coefficients have semantic meaning as follows: w_1 is the output when the input is low (x=0), and w_2 is the output when input is high (x=1). When the input value is high, the first expansion function has a value of zero, while the second has a value of one. This linear transform can be extended to characterize more than two input states. For example, if $\{w_1,w_2,w_3\}$ represents the output for {low, medium, high} input values, the expansion can be written $y = w_1 T(x-0) + w_2 T(x-0.5) + w_3 T(x-1)$. T(x) is a triangular expansion function given by MAX(0,1-|x/b|) where b is the triangle half-base. For n uniformly spaced centers, b is given by 1/(n-1). This expansion can also be extended to more than one input dimension, whereupon it becomes the multi-linear expansion. This extension is accomplished by forming all the products of expansion functions, one from each input dimension. The total number of expansion functions, H, is the product over all input dimensions, of the number of expansion centers in the input dimension. For the case of 3 input dimensions, where the input set is $\{x_1, x_2, x_3\}$, expanding on two centers in each dimension (at 0 and 1) gives a 2x2x2 multi-linear network with the expansion functions [8]

$h_1 =$	$(1-x_1)$	$(1-x_2)$	$(1-x_3)$
$h_2 =$	x_1	$(1-x_2)$	$(1-x_3)$
$h_3 =$	$(1-x_1)$	x_2	$(1-x_3)$
$h_4 =$	x_1	x_2	$(1-x_3)$
$h_5 =$	$(1-x_1)$	$(1-x_2)$	x_3
$h_6 =$	x_1	$(1-x_2)$	x_3
$h_7 =$	$(1-x_1)$	x_2	x_3
$h_8 =$	x_1	x_2	x_3

(1)

An expansion of each of three dimensions into “low”, “medium”, and “high” regions would give H=3x3x3=27 expansion functions. T(x) can be interpreted as a fuzzy membership function, describing how much an input parameter falls into a particular category, although the product formulation of the expansion functions differs from the traditional minimum formulation of fuzzy logic [19]. There are configurations in which the locations of the expansion function centers are adaptable [8], but these have not been pursued for this paper.

The net output is a linear combination of these functions of the input vector, with the coefficient of the jth expansion function being called w_j. The net output is

$$y = \sum_{j=1}^{H} w_j h_j = \vec{w} \cdot \vec{h} \qquad (2)$$

Vector notation will be used to represent any set of H quantities that correspond to the H expansion functions, such as the set of weights, **w**, connecting the second layer nodes to the output node, or the set of second layer node values, **h**. Any functional mapping (except one with an infinite number of discontinuities) can be represented by this type of expansion, as long as enough centers are used. A great advantage the multi-linear network has over nonlinear connectionist configurations like a multi-layer feed-forward network is that it can have a direct correspondence to a rule set. For example, the rule “IF (x_1 is low and x_2 is low and x_3 is high) THEN target value is low” could be implemented by setting the corresponding weight (w_5 in the above 2x2x2 network) equal to 0. Likewise, if a multi-linear net learns some set of weights, the corresponding knowledge can be directly extracted.

4. Supervised Training

The behavior of the connectionist controller is completely determined by the values of its weights. Supervised training is the process of adjusting the network weights until the response of the network mimics the response of the expert. A first approximation of the weights can be obtained by evaluating the expert response when the input matches one of the expansion function centers exactly. For example, as seen in Fig.1, the expert controller assigns a target priority of 0.82 to a target that is near, low, and on bore-sight (i.e. an input vector {0,0,0}). For this set of inputs, the first expansion node output, h_1, has a value of 1, while all the other nodes have a value of 0. If the first weight is set to 0.82, the network will give exactly the same response as the expert controller for this particular input state. Each of the H weights can be obtained, using the expert controller output values at the corresponding expansion function centers. The target priorities obtained at the centers of the 2x2x2 eight expansion function network are shown in Table 2.

j	x1	x2	x3	w[j]	
1	0	0	0	0.82	near/low/on-bore-sight
2	1	0	0	0.0	far/low/on-bore-sight
3	0	1	0	1.78	near/high/on-bore-sight
4	1	1	0	0.16	far/high/on-bore-sight
5	0	0	1	0.30	near/low/off-bore-sight
6	1	0	1	0.0	far/low/off-bore-sight
7	0	1	1	0.38	near/high/off-bore-sight
8	1	1	1	0.12	far/high/off-bore-sight

	near	far	near	far
high	1.78	0.16	0.38	0.12
low	0.82	0	0.30	0
	on-bore-sight		off-bore-sight	

Table 2. The weights assigned to the eight expansion function nodes of a 2x2x2 multi-linear network, obtained by evaluating the rule-based controller output at the expansion function centers, in list and tableau forms.

The performance of the multi-linear fire controller with these weights was evaluated as follows. A 2x2x2 network had its weights set to these values. The network was then used as the fire controller in a simulation of 10 sorties of 100 missiles each. The ABL with this fire controller intercepted 560 of the 1000 missiles. The 8 expansion function network is in essence interpolating on eight points that match the rule-based controller exactly. The network thus obtained does not perform as well as the rule-based controller itself (which intercepts 625 of the 1000 missiles) although it does perform better than random target selection (which intercepts 512 of the 1000 missiles.)

The multi-linear network can be made to match the expert to any desired accuracy by using enough expansion functions. Table 3 shows the performance of various sized multi-linear networks, with weights found by evaluating the rule-based controller on the expansion function centers. For example, when the ground range to target is expanded onto seven centers (i.e. very near, near, somewhat near, medium range, somewhat far, far, very far), and the target altitude and turret slew are expanded into seven and three centers, respectively, the resulting 147 expansion functions very nearly match the rule based controller performance.

H	configuration	intercepts
8	2x2x2	560
27	3x3x3	562
32	4x4x2	616
147	7x7x3	617
rule-based		625

Table 3. Performance of various sized multi-linear networks, initialized to match rule-based controller on expansion function centers, against 10 sorties of 100 missiles each.

In general, supervised training is an attempt to adjust the weights so that the network response best mimics a given controller over the whole input domain, rather than just at the expansion function centers. This is accomplished with the intermediary device of a training set, consisting of pairs of input vectors and the associated expert controller output. A training set can be obtained by collecting historical data, by polling experts, by experimental measurements, or as in this case, by a evaluating a knowledge-based controller over a set of input values.

The following index notation will be used, with no implicit summation over repeated indices:

$x_{ik} \equiv$ kth component of the ith training input parameter set

$\{x_{i1}, x_{i2}, ..., x_{iD}\}$ the ith training vector, representing a particular controller input

$t_i \equiv$ the expert controller output given the ith training vector as input

$h_{ij} \equiv$ the value of the jth expansion function produced by the ith training vector

$y_i \equiv$ the network output for the ith training vector

$w_j \equiv$ the jth weight of the multi-linear network

$D \equiv$ dimension of the input state vector

$H \equiv$ number of expansion functions

$N \equiv$ number of training set pairs

A baseline training set of 1000 vector - output pairs was generated, where each vector contains three parameters selected at random from ranges consistent with the simulated scenario, and each output is the corresponding expert controller target priority. The expansion functions of the network, such as those given in (1), are evaluated for each training vector, to get the h_{ij}. For a given set of weights, w_j, the network output for the i^{th} training vector is

$$y_i = \sum_{j=1}^{H} w_j h_{ij} \tag{3}$$

The mean square error, Q, for a particular set of weights and the given training set, is

$$Q = \tfrac{1}{N} \sum_{i=1}^{N} \left(y_i - t_i\right)^2 \tag{4}$$

Supervised training is a search for the set of weights that minimize Q. An efficient formulation for evaluating Q is obtained by expanding (4) and pre-summing over the training set:

$$Q = \tfrac{1}{2} \sum_{j=1}^{H} \sum_{k=1}^{H} w_j w_k A_{jk} - \sum_{j=1}^{H} w_j b_j + c \tag{5}$$

where **A** is a symmetric H by H matrix with elements $A_{jk} = \tfrac{2}{N} \sum_{i=1}^{N} h_{ij} h_{ik}$, and **b** is a vector of H elements, with elements $b_j = \tfrac{2}{N} \sum_{i=1}^{N} t_i h_{ij}$, and $c = \tfrac{1}{N} \sum_{i=1}^{N} t_i^2$. For H=27 expansion functions and a training set with N=1000 training pairs, it takes 8.48 seconds to compute **A**, **b**, and c, and then 0.010 sec for each evaluation of Q with (5) on a Macintosh Quadra.

5. Production training

In complex systems consisting of many interacting agents, the merit of the control strategy employed by one of the agents can be evaluated only by observing the system itself, or a simulation of the system. The merit of a controller is judged by how well it accomplishes its assigned goals as it interacts with other entities in a complex environment (i.e. in production). Production training differs from supervised training in that there is no given correct response for a given controller input vector. The production training process is a directed search for improved control strategies, in which trial controllers are generated from previous controllers, based on their performance evaluation. The performance of trial controllers is evaluated with the simulation package ELASTIC, described above. A performance evaluation based on simulation of 1000 engagements takes 90.0 sec on a Macintosh Quadra. A production evaluation of a controller performance requires 9000 times the computational resources of a performance evaluation of the kind used for supervised training.

With a faster computer, on-line adaptation becomes feasible. For example, this simulation-based performance evaluation takes 1.71 sec on a DEC alphaStation. If production training can discover better controllers by evaluating 500 trial controllers, this system would be able to adapt to changes that occur in the environment on a 15 minute time scale.

6. Multi-linear Regression

Supervised training of a multi-linear network can be accomplished with efficient and robust regression methods. This opens a variety of possible applications for multi-linear networks, in which rapid supervised training is more important than the non-linear capabilities of other connectionist architectures. These regressive methods produce the set of weights that exactly minimize the mean square error between the net and the training set. This exact solution is used in the assessment of the gradient descent and directed search methods that follow. The regressive methods are not otherwise relevant to production training.

The mean square error between the multi-linear network output and the training set for a particular set of weights was given in (4). The gradient of this mean square error is found by differentiating (4) with respect to weights. The j^{th} component of the gradient of the mean square error is

$$G_j \equiv \frac{\partial Q}{\partial w_j} = \tfrac{2}{N} \sum_{i=1}^{N} (\sum_{k=1}^{H} w_k \ \ h_{ik} - t_i) h_{ij} = \sum_{k=1}^{H} w_k A_{kj} - b_j \qquad (6)$$

In vector notation, the gradient of the mean square error is

$$\vec{G} = \vec{\ } Q = \vec{w} \ \ \overleftrightarrow{A} - \vec{b} \qquad (6a)$$

A and **b** are independent of the weights. The set of weights that minimizes the mean square error is found by setting each component of the gradient to zero. This gives a set of H equations known as the normal equations. The weight vector that minimizes Q is obtained by inverting the normal matrix, **A**.

$$w_j = \sum_{k=1}^{H} A_{jk}^{-1} b_k \qquad (7)$$

When the problem is well posed, **A** is a symmetric, positive definite matrix [20]. In this case, the matrix can be inverted by the very efficient Cholesky decomposition method. The Cholesky decomposition of **A**, and the vector of weights corresponding to a given **b**, are found using modified versions of the routines choldc and cholsl [17]. For the eight expansion function network, with the 1000 pair training set described above, the weights obtained by Cholesky inversion of the normal matrix are shown in Table 4. The optimal weight vector gives an rms error over the training set (i.e. the square root of Q) of 0.0736146. While this set of weights no longer matches the expert controller at the expansion function centers, it provides a better mimicry over the whole input domain.

	near	far	near	far
high	2.48	-0.50	-0.56	0.95
low	0.63	-0.57	0.18	-0.11
	on-bore-sight		off-bore-sight	

Table 4. The weights that minimize the mean square error of a 2x2x2 multi-linear network relative to the 1000 training pair training set, obtained by Cholesky inversion of the normal matrix.

A 2x2x2 network had its weights set to the values that minimize the error with respect to the 1000 pair training set (shown in Table 4). The network was then used in the fire controller in a simulation of 10 sorties of 100 missiles each. The ABL with this fire controller intercepted 586 of 1000 missiles. The 8 expansion function network which best mimics the rule-based controller does not perform as well as the rule-based controller itself (which intercepts 625 of the 1000 missiles) but it does perform significantly better than the 2x2x2 network which had its weights initialized on the eight expansion function centers.

If the normal matrix happens to be nearly singular, direct inversion is problematic [17,18,20]. In this case, singular value decomposition can be used to invert the normal matrix, **A**, or the design matrix (the N by H matrix with elements h_{ij}). The singular value decomposition of the design matrix, and the vector of weights corresponding to a given t_i, are found using modified versions of the routines svdcmp and svbksb [20]. The normal matrix can also be inverted by singular value decomposition. This method produces the same number of singularities as singular value decomposition of the design matrix. When the matrices are not singular, the same weight vectors are generated. When the matrices are singular, different weight vectors are obtained, but both give the same output when used in (2).

The relative computational requirements of these three regressive methods have been examined for the case of a 3x3x3 multi-linear network (H=27), with 1000 training pairs. The roughly 1.5 million multiply-and-adds needed to set up **A** and **b** require 8.48 sec on a Macintosh Quadra. Cholesky inversion of the 27x27 normal matrix then requires 0.066 sec, while singular value decomposition of the normal matrix requires 1.38 sec. 41.5 sec are required to set up the 1000x27 design matrix, and a further 36.7 sec to invert it by singular value decomposition.

7. Gradient Descent Training Methods

The regressive methods provide a direct solution for the set of weights that gives the multi-linear network that best matches the training set. In other widely used connectionist configurations, there is no relation like (7) that can be solved for the optimal set of weights, but there are explicit formulations for the mean square error, and for the gradient of the mean square error with respect to the weights, analogous to (5) and (6). Given a set of weights, the gradient can be used to generate a new set of weights that ought to have a better match to the training set. An adaptable controller can then be trained by an iterative generation of improved weight sets. Supervised training with any of several gradient descent methods (e.g. greedy hill climbing or conjugate gradient methods, with back propagation of errors used to find the gradient) make up the majority of neural network applications, and are well characterized [18]. Representative gradient descent methods were evaluated for supervised training of the multi-linear network on a training set produced with the knowledge-based controller. These methods are characterized by how many evaluations of the mean square error are required to reach a solution close to the optimal solution. Even though gradient descent approaches are not suitable for general production training, they can be used as a standard of comparison for the directed search methods that are suitable for production training.

7.1 Gradient descent with individually presented training pairs.

A common gradient descent supervised training method works as follows. One of the training pairs, say the ith one, is selected. For a given vector of current weights **w**, the square error between the network output for the ith training pair input, and the actual ith training pair output value is

$$q = \left(\vec{w} \ \ \vec{h} - t\right)^2 \tag{8}$$

The gradient of this square error with respect to the weights is

$$\partial q / \partial \vec{w} = 2(\vec{w} \ \ \vec{h} - t)\vec{h} \tag{9}$$

A second order Taylor series expansion of (8) about the current weight vector gives a quadratic approximation for the square error as a function of an increment to the weight vector:

$$q(\vec{w} + \delta\vec{w}) = q(\vec{w}) + \delta\vec{w} \ \ \vec{\ } q + \tfrac{1}{2}\delta\vec{w} \ \ \vec{\ }(\delta\vec{w} \ \ \vec{\ } q) \tag{10}$$

By expanding (10), using (8) and (9), and assuming that the weight vector increment is in the direction opposite to the gradient of q, the weight vector increment that gives a zero error value in (10) is easily found to be

$$\delta\vec{w} = \frac{\left(t - \vec{w} \ \ \vec{h}\right)\vec{h}}{h^2} \tag{11}$$

A multi-linear network using a weight vector formed by adding the increment in (11) to the original weight vector would give a good output value for the current training vector, but would degrade performance for other training pairs in the training set. This increment is therefore reduced by a factor known as the learning rate, η, so that some of the previous training is retained. The increment to the jth weight resulting from presenting the ith training vector is thus

$$\delta w_j = \frac{\eta \sum_{l=1}^{H} (t_i - w_l h_{il}) h_{ij}}{\sum_{k=1}^{H} h_{ik}^2} \tag{12}$$

In an epoch, all the training pairs in the training set are presented in sequence, and the weights are incremented after each presentation using (12). The learning rate is initially set to a value of 1.0, to rapidly get to the neighborhood of the solution, and then gradually reduced to provide for convergence. The annealing schedule by which the learning rate is reduced depends on the particular training set, and generally requires some trial and error adjustments. For the 1000 training pair set derived from the knowledge-based fire controller, a power law annealing schedule, in which the learning rate is reduced by a factor of 0.9998 after each training pair, was found to be effective.

This individually presented gradient descent method was used for supervised training of a 2x2x2 multi-linear network, using the 1000 pair training set described above. The eight weights were initialized to zeros, which gave a starting root mean square error of 0.4308. It took 27 epochs to reach a set of weights that produced a root mean square error of 0.07435 (within 1% of the optimal solution value of 0.073614 obtained by Cholesky inversion). This method thus requires an equivalent of 27000 trial weight vector evaluations, 27000 gradient evaluations, and 27000 estimates of the increment size. If it is supposed that the gradient could not be implicitly evaluated, it might be expected that the weight vector increment corresponding to each training pair would require eight evaluations of q to estimate the gradient, plus six or ten more evaluations to bracket the minima in the gradient direction. Application of this approach without the implicit gradient information of (9) would require on the order of 500,000 trial evaluations to reach within 1% of the solution. Note, however, that these are not evaluations of the performance of the controller over the whole input domain, but rather evaluations of the controller response at a single input state.

The benefit provided by initializing the weights to a good starting vector was then examined by starting with the weights initialized to the values of the expert controller on the expansion function centers (from Table 2). The training still required 27 epochs to reach a solution within 1% of the optimal solution. The weight vector gets to the right neighborhood within a few hundred training pair presentations, regardless of its starting point.

7.2 Batch mode gradient descent training.

An alternative gradient descent approach uses the entire training set at once to generate a new trial weight vector. The mean square error over the whole training set is given in (4), while the gradient of this mean square error (with respect to the weights) is given in (6). As in the individually presented training pair case, the mean square error can be expanded about the current weight vector into a second order Taylor series. The various terms in the Taylor series can be expressed in terms of **A**, **b** and **w**, again constraining the weight increment to the direction opposite the gradient. The weight increment that minimizes the second order expansion of Q is found to be

$$\delta\vec{w} = \frac{-G^2\vec{G}}{\vec{G}\ \overset{\leftrightarrow}{A}\ \vec{G}} \tag{13}$$

Again, a learning rate is used to aid convergence, and the weight increment in index notation becomes

$$\delta w_j = \frac{-\eta G_j \sum_{k=1}^{H} G_k G_k}{\sum_{l=1}^{H} \sum_{m=1}^{H} G_l A_{lm} G_m} \tag{14}$$

This batch gradient descent method was used for supervised training of the 2x2x2 multi-linear network, using a learning rate of 0.9. The weights were initialized to the rule-based values on the expansion function centers. The initial rms error between the multi-linear net and the training set output values was 0.21645. This method required 49 training epochs to reach a solution with an rms error of 0.07435 (i.e. 1% more than the least rms error solution). If the weights are instead initialized to zeros, this batch gradient descent with implicit increment size estimation requires 59 epochs.

The equivalent number of fitness evaluations (i.e. evaluations of Q) can now be estimated to characterize cases in which the gradient and estimated weight increment size are not available implicitly. Assuming that H additional evaluations of Q would be required to estimate the gradient, and again allowing six to ten evaluations to bracket the minima in the direction of the gradient, gives that these 49 epochs would require the equivalent of around 800 evaluations of Q for this batch gradient descent method to find a solution within 1% of the best solution. In this batch approach, each evaluation reflects the performance of the controller over its entire domain.

8. Directed Search Training Methods

In directed search methods, new trial solutions are generated from previous trial solutions and information about the performance of those previous trial solutions. Directed search methods can be used for supervised training. In production training, where gradients may not be well defined and where there may be many local optima, directed search methods might be the only means of improving adaptable controllers. Three directed search methods have been examined: pivot and forward-biased random offset with simulated annealing, downhill simplex, and the genetic algorithm. These methods are compared with the gradient descent approach for supervised training. They are also characterized for production training.

8.1 Directed search by pivot and forward-biased random offset.

The pivot and random offset method starts with a weight vector (the pivot), and evaluates the performance of the corresponding multi-linear network controller. This performance is given by the mean square error relative to a training set for supervised training (i.e. by evaluation of (5)), and by a simulation-based performance evaluation for production training. An offset is obtained by applying a random increment to the pivot weight vector, and the performance of the offset is evaluated. If the performance is improved, the offset weight vector is accepted as the new pivot. Otherwise, a new offset is tried. When a successful offset is found, the next trial weight vector offset is biased toward the same direction.

The weight vector increment consists of a forward biasing component plus a random increment to each of the H weights. The forward biasing component is equal to one half of the last successful increment. The random component is selected from a uniform distribution in the interval $(-\omega, \omega)$, where ω is implemented as an adaptive parameter. The algorithm for adjusting ω increases ω by 2% when the offset performance exceeds the pivot performance, and decreases ω by 2% when the offset performance is worse than the pivot performance. If ω becomes too large (>0.5) or too small (<0.001), it is reset to an intermediate value (0.05).

The performance of this pivot-offset method is stochastic, depending on the initial random seed used to generate the random weight increments. For supervised training in the 8 expansion function case with 1000 training pairs, with the weights initialized to zeros, a series of 10 independent pivot-offset searches required as few as 317 performance evaluations, or as many as 2002 to converge to a solution with rms error within 1% of the error of the optimal solution. The mean requirement for the 10 searches was 1152 evaluations. This compares to the equivalent of about 800 evaluations required to reach this accuracy with the batch gradient descent method. If the initial set of weights is set to the values obtained at the corresponding expansion function centers with the knowledge-based controller, the required number of performance evaluations ranges from 278 to 1392, with a mean (in 10 runs) of 920. A good starting weight vector improves the search efficiency.

A simulated annealing approach has been implemented to allow escape from local optima that typically arise in production training. The offset is always accepted if its performance exceeds that of the pivot. With simulated annealing, the offset is also occasionally accepted when its performance is worse. The probability of acceptance of a worse solution depends on a “temperature” parameter, T, which has the same units as the measure of performance. The implementation of the acceptance criteria is: in a search for increased performance, if the performance of the offset exceeds the performance of the pivot plus the temperature times the natural log of a uniform random variate between 0 and 1, the offset is accepted as the new pivot. In a search for smaller measure of performance (e.g. search for smaller Q), if the measure of performance of the offset is smaller than the measure of performance of the pivot minus the temperature times the natural log of a uniform random variate between 0 and 1, the offset is accepted as the new pivot. This produces a series of trial controllers that has a Boltzmann distribution of performance [17, 18, 21]. The temperature is reduced by a constant factor after each performance evaluation (producing a power law annealing schedule), gradually reducing the likelihood of accepting a worse offset. Simulated annealing degrades supervised training of a multi-linear network, because there is only one minima.

This pivot-offset method was applied to production training, using simulation of 10 sorties with 100 missiles each to evaluate the trial multi-linear network fire controllers. A 2x2x2 multi-linear network was used, with weights initialized to the expansion function center values of the shortest time-to-kill knowledge-based controller. The initial temperature was set at 5 missile kills. The annealing schedule reduced the temperature by 0.975 after each performance evaluation, so that by the 150th evaluation, worse offsets were rarely accepted. The pivot-offset directed search found a weight vector that intercepted 667 of the 1000 missiles, after evaluating 500 trial weight vectors. This is significantly better than the performance of the original knowledge-based controller. The weights of the adapted controller are shown in Table 5. In comparison with the original controller, characterized in part in Table 2, the adaptable controller has learned that better performance is obtained by reducing the priority of low targets in general, except far, off-bore-sight ones, and increasing the priority of high, near, off-bore-sight targets.

	near	far	near	far
high	2.27	0.06	1.27	0.04
low	-0.27	-1.03	-0.33	0.61
	on-bore-sight		off-bore-sight	

Table 5. The weights assigned to the eight expansion function nodes of a 2x2x2 multi-linear network, obtained by a forward biased pivot-offset search, with simulated annealing, evaluating 500 trial sets of weights.

8.2 Directed search by Downhill Simplex.

Instead of tracking two weight vectors (pivot and offset), the simplex method uses H+1 weight vectors which form a “simplex” in the H dimensional weight vector space. In an iterative process, the worst corner of the simplex (that weight vector which gives the worst controller performance) is moved to the “opposite” side of the simplex, or failing that, in “toward” the center of the simplex. For supervised training, the performance of a weight vector is the mean square error relative to the training set, evaluated with (5). The downhill simplex search is performed with modified versions of the routines *amoeba* and *amotry* [17]. This method requires about the same computational time for each performance evaluation as the pivot-offset method.

The number of evaluations required to attain a solution with an error within 1% of the error of the optimal solution was observed to be a sensitive (in fact, chaotic) function of the starting simplex vertices. One vertex was taken as the origin (all weights set to zero). The other H vertices of the initial simplex had one of the H components incremented by δ, and the rest unchanged, where δ is an adjustable parameter. For 10 different values of δ, ranging from 0.2 to 0.4, the number of required evaluations ranged from 226 to 572, with an average requirement of 377 evaluations. This compares to 1152 for the pivot-offset method. If the initial simplex is constructed from a set of weights equal to the true values obtained at the corresponding expansion function centers, the mean required number of performance evaluations (averaged over 10 runs) is 305.

This downhill simplex method was applied to production training, using simulation of 10 sorties with 100 missiles each to evaluate the trial multi-linear network fire controllers. A 2x2x2 multi-linear network was used, with weights initialized to the expansion function center values of the shortest time-to-kill knowledge-based controller. The downhill simplex directed search found a weight vector that intercepted 676 of the 1000 missiles, after evaluating 500 trial weight vectors. This is significantly better than the performance of the original knowledge-based controller. The weights of the adapted controller are shown in Table 6. This set of weights is distinct from that found by the pivot-offset search. It was verified that this set of weights is a locally optimal solution, by evaluating the performance of 16 slightly displaced weight vectors (each having one of the 8 weights perturbed by a small positive or negative increment). The resulting controller was also validated by evaluating the performance of the solution against a completely independent set of 10 sorties of 100 missiles each. In this test, the original rule-based controller intercepts 634 missiles, while the multi-linear net obtained by the downhill simplex search intercepts 678 missiles.

	near	far	near	far
high	2.64	0.51	0.80	0.03
low	-0.13	0.43	0.35	0.45
	on-bore-sight		off-bore-sight	

Table 6. The weights assigned to the eight expansion function nodes of a 2x2x2 multi-linear network, obtained by a downhill simplex search, evaluating 500 trial sets of weights.

8.3 Directed search with genetic algorithms.

A third directed search approach, evolutionary programming, is based on an analogy with biological evolution [22]. A set of trial solutions, called the population, evolves from generation to generation. The set of weights that characterize a trial multi-linear network controller makes up the chromosome associated with an individual. Each weight is a gene. In this application, the gene can take any real number value. Mutation of a gene is accomplished by adding an increment which is selected from a uniform random distribution in $(-\omega, \omega)$. Two individuals can be used to breed a third trial individual, using two point cross-over: the descendent receives the first and last parts of its chromosome from one parent, and the middle part from the other parent, with a random location of the cross-over points.

The members of the first generation of trial controllers are created by mutating an original set of weights. The measure of performance, or fitness, of each of the trial controllers in the population is then evaluated. The worst half of the population is discarded, to be replaced by new individuals. A new individual is generated from two parents, which are selected according to their fitness. The selection of parents is based of rank order of fitness, where the best controller in the population is five times more likely to be selected as a parent than the worst of the remaining top half of the population. The chromosome resulting from crossing two parents is then mutated. When the discarded half of the population has been replaced by new individuals, the performance of these new individuals is evaluated. This process is then iterated over successive generations.

For supervised training, the performance of a weight vector is the mean square error relative to the training set, evaluated with (5). The population size was set to 40, so that 20 new trial solutions are generated and evaluated in each generation. An annealing process was used for the mutation rate. Initially, after a new individual is formed by crossing parents, each weight of the new chromosome is incremented with a maximum increment of 0.1. The maximum increment is reduced by a factor of 0.99 after each generation, so that less and less mutation is occurring as the search converges to a solution. This genetic algorithm search was applied for a 2x2x2 multi-linear network, using the 1000 pair training set described above. The initial population was obtained from mutants of the weights obtained by applying the original rule-based controller at the eight expansion function centers (Table 4). 28 independent searches were undertaken. The worst of these required 6160 trial evaluations to reach a solution within 1% rms error of the optimal solution, while the best required 1620. The average requirement was 2564 performance evaluations.

This genetic algorithm was applied to production training, using simulation of 10 sorties with 100 missiles each to evaluate the trial multi-linear network fire controllers. A 2x2x2 multi-linear network was used, with weights initialized to the expansion function center values of the shortest time-to-kill knowledge-based controller. The genetic algorithm directed search found a weight vector that intercepted 678 of the 1000 missiles, after evaluating 500 trial weight vectors. This is significantly better than the performance of the original knowledge-based controller. The weights of the adapted controller are shown in Table 7. This set of weights is distinct from that found by the other searches. It was verified that this set of weights is a locally optimal solution. The new solution was validated by evaluating its performance against a completely independent set of 10 sorties of 100 missiles each. In this test, the original rule-based controller intercepts 634 missiles, while the multi-linear net obtained by the genetic algorithm search intercepts 682 missiles.

	near	far	near	far
high	2.48	-0.01	-0.34	1.33
low	-0.17	-0.10	0.47	-0.27
	on-bore-sight		off-bore-sight	

Table 7. The weights assigned to the eight expansion function nodes of a 2x2x2 multi-linear network, obtained by a genetic algorithm search, evaluating 500 trial sets of weights.

8.4 Demonstration of automatic adaptation to environmental changes.

A 4x4x2 multi-linear network (4 expansion centers in ground range to target and target altitude, 2 in slew angle) was constructed and initialized by setting the weights equal to the shortest time-to-kill controller output. As shown in Table 3, a fire controller using this network was able to intercept 616 missiles of the 1000 missiles in simulation of 10 sorties of 100 missiles each. The multi-linear network output is shown in Fig. 2 for this initial set of weights, as a function of x_1 (normalized range to target) and x_2 (normalized target altitude), for $x_3 = 0$ (target on bore-sight). Comparison with Fig. 1 shows how well this H=32 multi-linear network mimics the original controller, at least on bore-sight.

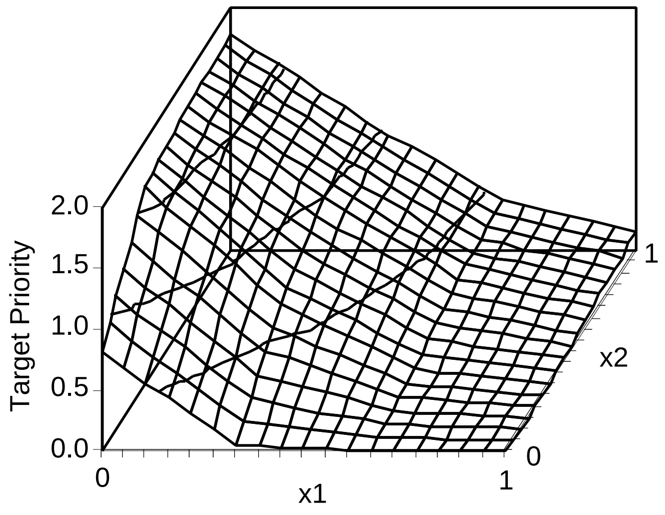

Figure 2. The output of a 4x4x2 multi-linear network fire controller, as a function of x_1 (normalized ground range to target) and x_2 (normalized target altitude), for x_3=0 (target on bore-sight), where the weights are set to the shortest time-to-kill values on the expansion function centers.

This multi-linear network fire controller was then evolved using the genetic algorithm directed search, using simulation of 10 sorties with 100 missiles each to evaluate the trial multi-linear network fire controllers. After evaluating 500 trial controllers, a new controller was discovered that was able to intercept 684 of the 1000 missiles. Further search (5500 trial evaluations) found a new controller that intercepted 690 of the 1000 missiles. Additional search (to 75,420 evaluations) found no further improvement. The output of this new controller is shown in Fig. 3, again as a function of x_1 and x_2, for $x_3 = 0$.

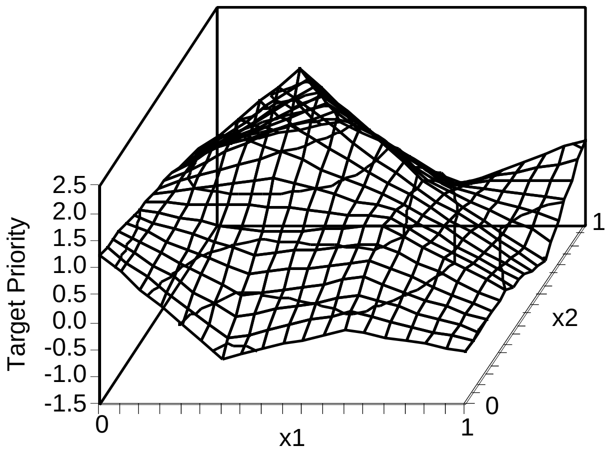

Figure 3. The output of a 4x4x2 multi-linear network fire controller, as a function of x_1 (normalized ground range to target) and x_2 (normalized target altitude), for x_3=0 (target on bore-sight), after directed search for improvements.

The above fire-controller has optimized its behavior for a scenario with 100 medium-range (400 to 600 km) missiles per sortie. The environmental conditions were then changed to the scenario in which a sortie consists of 50 medium-range missiles and 50 short-range (250 to 350 km) missiles. These short-range missiles burn out at a lower altitude, and at an earlier time, making them harder to intercept. The contingency for defense against short range missiles had in no way been pre-programmed into the controller. The traditional approach to adapting the behavior of a knowledge-based controller is to assemble a group of experts to add knowledge to the controller to prepare it for the new environmental conditions. The multi-linear network controller with simulation-based directed search adaptation allows the controller to automatically adapt its behavior to the new conditions.

When the multi-linear network controller adapted to the medium-range missile only scenario is applied to the new scenario with medium and short range missiles, it is able to intercept 581 of 1000 missiles. The genetic algorithm method was used to adapt the controller to the new scenario. Using 600 trial controller evaluations (requiring less than 15 minutes to compute on a DEC alphaStation), a new controller was automatically discovered that intercepted 606 of the 1000 missiles. The output of this new controller is shown in Fig. 4, again as a function of x_1 and x_2, for $x_3 = 0$. Comparison with Fig. 3 shows that the controller has adapted to the presence of short range missiles by giving increased priority to lower altitude targets.

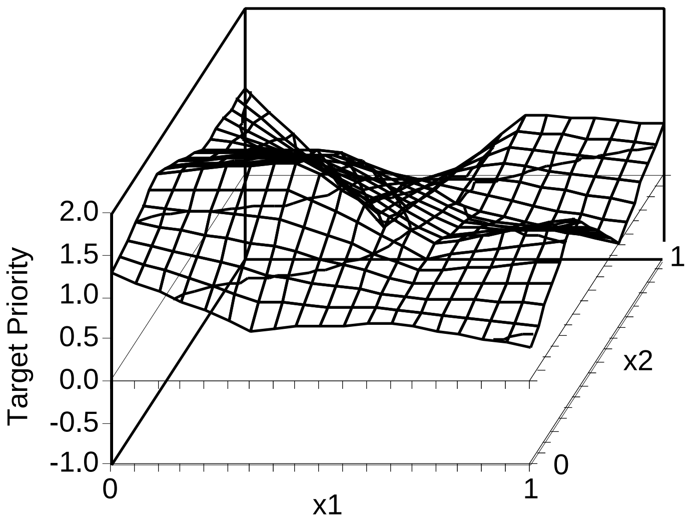

Figure 4. The output of a 4x4x2 multi-linear network fire controller, as a function of x_1 (normalized ground range to target) and x_2 (normalized target altitude), for x_3=0 (target on bore-sight), after directed search for improvements with a modified scenario that includes short-range missiles.

The battlefield environment again change, so that instead of having the missiles launched over a 3 minute period, 100 medium-range missiles are launched over a 5 minute period. The shortest time-to-kill controller gives 671 intercepts of 1000 missiles for this new scenario. A genetic algorithm directed search found a 4x4x2 multi-linear network controller that gives 704 intercepts, after evaluating 500 trial weight vectors. Again, the adaptable controller automatically discovers a new controller behavior that is well adapted to un-anticipated environmental conditions.

A third modified scenario had 25 simultaneous launches, from a 10 by 10 km launch zone, located 300 km in front of the loiter box. The performance of trial controllers is evaluated by simulating 40 independent sorties, again giving 1000 total missile engagements. The shortest time-to-kill controller gives 337 intercepts of 1000 missiles for this scenario. A genetic algorithm directed search found a 4x4x2 multi-linear network controller that gives 346 intercepts, after evaluating 500 trial weight vectors. It is not surprising that so little improvement can be attained for this scenario, since there is little to distinguish the 25 simultaneously launched missiles.

9. Conclusion

A simulated battlefield with an airborne laser missile interceptor provides an excellent test-bed for investigating controllers of entities that interact with a complex system. An approach has been investigated for enabling the controller to adapt to its environment. The approach is to transform a knowledge-based controller into an adaptable connectionist representation, use supervised training to initialize the weights so that the adaptable controller mimics the knowledge-based controller, and then use directed search with simulation-based performance evaluation to find controllers that are better adapted to the environmental conditions. The approach has the added benefit that the new knowledge can be directly extracted from the automatically discovered controllers.

The process of production training brings two new capabilities to the controller. First, it enables an automated search for improvements over the baseline expert controller. In a scenario in which a poor fire controller gives 512 intercepts, and the state-of-the-art knowledge-based fire controller gives 625 intercepts, directed searches automatically discovered new controller behaviors that give as many as 690 intercepts.

Three directed search methods were examined: pivot with forward-biased offset, downhill simplex, and evolutionary programming. These methods require a means of evaluating the performance of trial controllers, but do not require information about the gradient of the performance with respect to the adaptable controller parameters. They have the ability to avoid becoming trapped at poor but locally optimal controllers. All three methods successfully discovered improved controllers. Each of the improved controllers were verified to be locally optimal. The three controllers were nevertheless quite distinct from each other, and from the original representation of the knowledge-based controller. The controllers discovered by directed search were validated by evaluating their performance against independent sorties.

The second capability brought by production training is that the connectionist controller can develop adaptations to changes in the rest of the system. These adaptations can be developed off-line, by production training against alternative scenarios. Real-time or on-line adaptation can be accomplished when the directed search process is carried out faster than the time scale of the changes in the system. Some examples of rapid, automatic adaptation to modifications in the environment were examined. Without this directed search capability, experts would have to be assembled to revise rule-based controllers in response to changes in the environment.

The supervised training process was found to be an important antecedent to the directed search production training process. This is because the directed search methods have a number of adjustable parameters (mutation rate, step size, variability in the initial set of trial controllers, temperature or mutation rate annealing schedule, etc.) Good values for these parameters can be determined during supervised training, since the performance evaluations used in supervised training are much faster than those used in production training.

The problem space (the set of all possible input states combined with the set of all attainable rule set mappings) was sufficiently large to demonstrate the validity of the methodology. More dramatic improvements in performance can be obtained by using this methodology with more elaborate rule sets and state representations. For example, including a fourth controller input that indicates the extent to which a given missile is clustered with other missiles could allow an automatic search for controllers that reduce the time wasted on slewing the turret.

Nothing in this process is specific to fire controllers, except the representation of the state and the rule set mapping. This methodology of transforming a knowledge-based controller to an adaptive structure, evolving the weights automatically in the context of a synthetic environment, and transforming back to an improved rule-set, should apply to a wide variety of systems.

References

[1] M. M. Waldrop, Complexity, Simon & Schuster, NY , (1992).

[2] H. Plotkin, Darwin Machines and the Nature of Knowledge, Harvard University Press, Cambridge, MA, (1994).

[3] A. Newell, Unified Theories of Cognition, , Harvard University Press, Cambridge, MA, (1990).

[4] H. A. Simon, The Sciences of the Artificial, MIT Press, Cambridge, MA, (1981).

[5] J. H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, (1992).

[6] Handbook of intelligent control: neural, fuzzy, and adaptive approaches, D. A. White and D. A. Sofge, eds., Van Nostrand Reinhold, New York, (1992).

[7] C. Barrett, R. Jones, U. Hand, "Adaptive Capture of Expert Knowledge," Los Alamos National Laboratory Technical Report LA-UR-95-1391(1995).

[8] G. Danczyk, S. Mortenson, W. Sailor, P. Stroud, "Airplane based free electron laser concept overview," proc. PBFEL Workshop, Los Alamos National Laboratory, (Dec. 4, 1990).

[9] "Teams to Submit Bids For Airborne Laser," Aviation Week & Space Technology, (June 10, 1996).

[10] P. D. Stroud, "Anisoplanatic, Bandwidth, and Scintillation Effects of Atmospheric Turbulence on Laser Propagation across Long Paths Between an Airborne Laser and a Flying Target", proc. CLEO, (1992).

[11] P. D. Stroud, "Anisoplanatism in adaptive optics compensation of a focused beam with use of distributed beacons," J. Opt. Soc. Am. A/Vol. 13, No 4, pp 868-874, April 1996.

[12] P. D. Stroud, "Statistics of intermediate duration averages of atmospheric scintillation," Optical Engineering, Vol. 35, No. 2, pp 543-548, Feb 1996.

[13] "TEMPEST (TACCSF Exploratory Model of Performance, Strategy, and Tactics)," numerical simulation package copyright by S. Mortenson, Los Alamos National Laboratory and the Regents of the University of California, 1992.

[14] ISSAC-ABEL, simulation package developed by W.J.Shaeffer and Associates.

[15] The Extended Air Defense Simulation (EADSIM) User's Reference Manual, US Army SSDC, Huntsville Alabama, 1993.

[16] J. Brown, C. Heydemann, J. Soukup, "Theater Air Command and Control Simulation Facility ABL Test 6," Airborne Laser Program report, Phillips Laboratory, Albuquerque NM, 1994.

[17] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C, 2nd ed., Cambridge University Press,(1992).

[18] T. Masters, Practical Neural Net Recipes in C++, Academic Press, Inc, Boston (1993).

[19] B. Kosko, Neural Networks and Fuzzy Systems, Prentice-Hall (1992).

[20] S. Ergezinger, E. Thomsen, "An Accelerated Learning Algorithm for Multi-layer Perceptrons: Optimization Layer by Layer", IEEE trans. Neural Networks, vol 6, **1**,(Jan 1995).

[21] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, "Equation of State Calculations by Fast Computing Machines," J. Chem. Phys., vol. 21, no. 6, (1953).

[22] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Co., (1989).